From Databases to Graph Visualization

Frédéric Gilbert, David Auber

From Databases to Graph Visualization

Frédéric GILBERT
LaBRI Université de Bordeaux
Gravité Inria Bordeaux Sud-Ouest
Bordeaux, France
frederic.gilbert@labri.fr

David AUBER
LaBRI Université de Bordeaux
Gravité Inria Bordeaux Sud-Ouest
Bordeaux, France
david.auber@labri.fr

***Abstract*—The first step of any information visualization system is to enable end user to import their dataset into the system. However, non expert user are faced to the difficult task of choosing how their data should/could be transform to be used in these Infovis systems. In that paper we address the case where end users want to use dataset in tabular format. We propose a novel method for automatic graph generation from these datasets. That method consists in first building taxonomy of dimensions. Then, that taxonomy is used to provide to user a system that enables to interactively navigate into the set of possible data transformation.**

***Keywords*-Database visualization, database analysis, automatic presentation, graph generation.**

I. Introduction

Nowadays, it is extremely easy to collect data from different sources. There are a lot of methods for collecting and storing data. Concerning the storage, databases are the most used way and contain basic data as words, numbers, date, time ... They are stored, in order to keep some informations as: who/what, when, where, how many time and so on ... But as the collect and the storage of data is easier, companies and research project are producing large databases. For instances, genome researchers collect data in order to understand how genes interact and commercials companies collect data in order to find customer behaviours (patterns) or outliers.

As the amount of data grows rapidly graphical visualization becomes necessary to understand results of users queries on databases. Projects as DEVise [1], Polaris [2] and Tableau [3] propose user interface for these tasks. They propose to follow the well known exploratory analysis process defined by: first an hypothesis, second an experiment and finally a discovery. But, with massive databases the number of possible experiment for a given hypothesis grows with the number of dimension contain in the database. Thus, end users are faced to the problem of the exploration of all these possibilities.

Let consider a geographer which have spent several years collecting information about air-plane traffic. For each fly the details collected are: the date of departure, the time of departure, the starting airport name, city and country, the fly company name , the aircraft type, the number of passenger in the fly, the day of arrival, the time of arrival and the ending airport name, city and country. in that case. The geographer may be able to find that some companies have got fly only between two airports or that between the airport A and B there is only fly in the morning. However, if the number of fly is huge, even these simple tasks will become difficult. Furthermore, finding more complex patterns in this dataset will become time consuming or completely impossible.

Using as input data table extracted from database, in that paper we propose to generate automatically the weighted graphs (set of entities and relations with attributes) one can find in these tables. We also provide a system that enable end user to visually explore that set of graphs. The contribution of that paper is a method that simplify database and highlight relations between entities. We assume that the proposed method make the exploration process more efficient and thus ease the user task. Our simplification method is based on the building of a taxonomy of the database dimensions which highlight nested dimensions in the dataset. Then using this taxonomy, we are able to highlight relations between entities and to generate simplified visualizations of the dataset.

This paper is structured as follow. First we present previous works. Then we describe our simplification method, and give an overview of our interactive system. We conclude with a case study applied on a manually generated dataset.

II. Previous work

According to user center of interest, we aim at providing simplified visualization of automatically generated graphs. In the following, we present previous work on data simplification and user interaction for automatic generation of visualization.

A. Data simplification

Data simplification is often associated to Principal Component Analysis [4]. But these methods which try to aggregate several dimensions to only one, works on quantitative values. In our work, we are looking for relations between entities and this kind of relations are hidden in nominal values. So we have to look also to Latent Semantic Analysis [5] which summarizes equivalent methods for nominal data.

B. User interaction in automatically generated visualization

As described in [6], [3], [7], automatic generation of a visualization according to datasets is one of the information visualization community's problematic. One can distinguish three kinds of approaches. The first one consist in asking users to build what they want to visualize, the second asks experts to build the visualization, the third one simplify the user interaction and propose automatic generation of visualization and the last one propose a notation for easily describe and manipulate hierarchical visualization.

For Smith's nodeXl software [6], the idea is to work with spreadsheet. In this case users will be able to compute things using formulas as usual with spreadsheet. And then use the computed values as attributes for entities and also for relations. They propose a tool which allow users to build a graph from a spreadsheet. The graph must be described by an edge list. The spreadsheet must contain at least two columns, and each line will describe the two entities that are in relation. So this tool provide an automatic method that avoids the use of a programming language and by the way allows no expert user to produce visual analysis of relational data. In that method due to the manual creation of the edges list, one cannot assume that an non-expert user will be able to construct an valid graph.

Now, according to ManyEyes [8], it is important to have multiple people who can work on the same data and discuss the visualization generated from this data set. They explain that it is difficult to give the good way to visualize a kind of data set. Maybe for two datasets differing from only one dimension, the way to visualize them will be totally different. That is why they have developed "Many Eyes", a web service that allows users to access to datasets, upload their own datasets and create visualizations. But this is only the first part of the process, the second one consists in proposing the visualizations generated in a discussion form in order to take profit of expert opinion. So instead of trying to build the "good" visualization for a data set, they prefer offer the opportunity to have a visualization build by several experts.

In Mackinlay's Tableau Software [3], one can load easily data, manipulate them and create visualization combining data's dimensions. The power of this tool is that it can choose for the user how to visualize efficiently the result of the dimensions' combination. An interesting part of this tool is the fact that by combinatorial test(called "Affinity"), dimensions will be automatically sorted in order to simplify the result. This is important because sometimes users cannot see that in huge databases. However, the "Affinity" combinatorial test is not described.

In [7], Slingsby et al. define a notation for describe hierarchical visualizations. With this notation user can easily encode layout themselves. Modifying the order of layout or the levels of the hierarchy allow users to produce easily visualizations. But nothing guide the user in order to construct correct and significant hierarchy. Furthermore the author only work on tree map [9] visualization for hierarchy and do not discuss about how to use their method on other dataset.

III. Data Analysis

We work on nominal database. We consider each value stored in the database as a nominal one. However our goal is to highlight relations between entities and not relations between dimensions. To simplify the data, we have considered using Latent Semantic Analysis [5]. These methods cause a loss of information that makes it unusable for the purpose of our work. In fact these methods try to reduce the number of dimension contained in the database. Our goal is to simplify the data without having to merge or push aside dimensions. So, instead of trying to find similarities between dimensions, we try to find relations between the dimensions and more precisely hierarchical relations. We already know that data contain three types of data as define in [10]: entities, relations and attributes. Entities are the studied objects, relations give information between entities and each one can have a lot of other information called attributes. Here, we consider that databases are tables where a row describes an entity and each value of this row is an attribute of this entity. We try to highlight relations in this table, finding significant matching between attributes of a same dimension (column) of the table. Then in order to visualize these relations, we decide to work with graphs visualized as node-link diagrams. In the graph, a vertex (node) represents an entity (row) of the table, and if two entities are in relation we add an edge (link) between them in order to materialize the fact that they are in relation. Our data analysis has two steps. First, we build a taxonomy of the table's dimension, then we look for relations between entities.

A. Taxonomy of dimensions

We describe here the way we simplify the data without loosing information. Our idea is to detect dimensions which have more importance than other in order to afterwards find relations between entities. That's why we decide to look for hierarchical relation between dimensions.

Notation: Let T be a table with m rows and n dimensions. We note by T_i the i^{th} dimension of the table and by Σ_i the alphabet associated to that dimension. $T_{i,j}$ is the j^{th} value of the i^{th} dimension of the table and $\Sigma_{i,j}$ is the j^{th} value of the i^{th} alphabet.

Definition: Consider T_i and T_j two dimensions of the table and their alphabet Σ_i, Σ_j. $\forall k \in [1, |\Sigma_j|]$ if all elements of T_j which have for value $\Sigma_{j,k}$ have in T_i the same value $\Sigma_{i,x}$ then T_i is ranked over T_j and we note $T_i \geq T_j$.

Continent	Country	City	Street	Name
Europe	France	Paris	Louis Pasteur	Dupont
Europe	France	Paris	Champs-Elysée	Durand
Europe	France	Bordeaux	Louis Pasteur	Martin
Europe	Germany	Berlin	Max Planck	Muller
Europe	Germany	Munich	Max Planck	Fischer
Europe	Spain	Madrid	Louis Pasteur	Fernandez
North America	USA	New York	Grand	Smith
North America	USA	Boston	Beacon	Do
North America	Canada	Calgary	Grand	Wilson

Table I
EXAMPLE OF AN ADDRESS TABLE

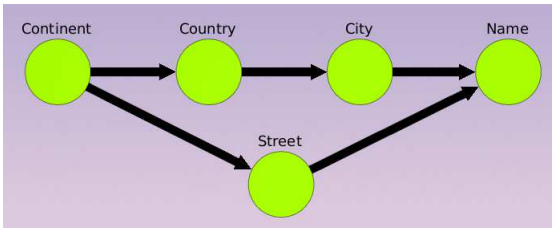

Figure 1. The taxonomy obtained from Table I. There are two branches: "Continent", "Country", "City", "Name" and "Continent", "Street", "Name". If there is an edge from node A to node B means that A is ranked over B. B give more precise information than A.

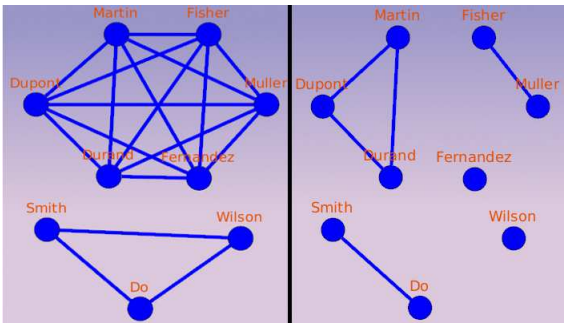

Figure 2. The left part represents the graph obtained using the Continent dimension. People who live in the same continent are connected. There are two connected component: the top one for Europe, the bottom one for North America. The right part represents the graph obtained using the Country dimension. People who live in the same Country are connected. There are five connected component, one for each country. As "Country" dimension is ranked under "Continent" dimension in the taxonomy, each connected component in the right part graph is an induced subgraph of the corresponding connected component in the left part.

Exemple: Consider the Table I, where each row contains information about peoples' address. The dimensions are: Continent, Country, City, Street, Name. Our method give us that "Continent" $\geq$ "Country" $\geq$ "City" $\geq$ "Name" and also that "Continent" $\geq$ "Street" $\geq$ "Name". We can"t have a hierarchy of the five dimensions because Louis Pasteur street and Grand Street deny relations between "City" and "Street" and between "Country" and "Street".

Property: Let consider two dimensions T_i and T_j. If we have $T_i \geq T_j$ and $T_j \geq T_i$, that means that T_i and T_j are equivalent.

According to the previous Property, if we have two equivalent dimensions, then we can consider only one of these dimensions for the rest of the task. And we assure that considering only one of the dimensions don't remove information.

Comparing dimensions, we are able to simplify the data without loosing information, we can order them and construct hierarchies of dimensions. These hierarchies can be merged into a single taxonomy of dimensions. In this taxonomy, top level dimensions are the more global ones and the bottom level dimensions are the more specific ones. But as we compare each dimension with the others, the taxonomy have non necessary edges. In our example: "Continent" $\geq$ "Country" $\geq$ "City". But as we compare all the possible pairs of dimensions we also have that "Continent" $\geq$ "City". And this edge will not give us much information in the taxonomy, so we have to clean the taxonomy removing all this kind of edges.

Property: Let consider a taxonomy of dimensions build using the above Definition. This taxonomy can't hold strong connected components.

In fact, if we have $T_1 \geq T_2 \geq T_3 \geq T_1$ that mean that we have a cycle in the taxonomy. This cycle is a strong connected component. Considering $T_1 \geq T_2 \geq T_3$, we have $T_1 \geq T_3$ and as we also have $T_3 \geq T_1$, we have that T_1 is equivalent to T_3. By extension we have $T_1 \geq T_2 \geq T_1$, so T_1 is equivalent to T_2. So finally we have T_1 is equivalent to T_2 which is equivalent to T_3. With a proof by recurrence, it's easy to generalise the result to strong connected component. So each time that we have a strong connected component in the taxonomy, that means that all the dimension of the strong connected component are equivalent. And the taxonomy can be simplify this way.

B. Relation between entities

In order to know if two entities are in relation, we check if they have the same value in, at least, one dimension. For example, let consider a table of planes, for each plane we have the name of the company controlling it. If two or more planes have the same value of controlling company, we have detect that these planes belong to the same company. This way we are able to define a belonging relation.

In order to avoid all pairwise comparisons, we will look for relations according to the taxonomy. We only search relation in the bottom (more specific) dimensions of the taxonomy. Because, if there is a relation according to a dimension, due to the taxonomy of dimension we are sure that these elements will be in relation in upper level of the taxonomy. But doing things this way, we will have a lot of edge in the graph.

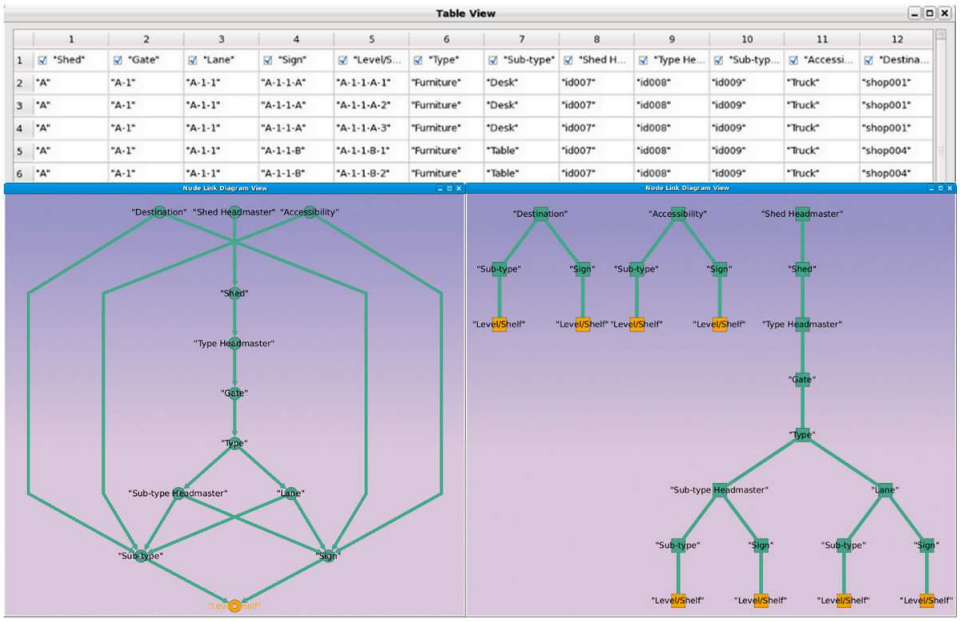

Figure 3. The ”Table View” window display the data. The first line holds the name of the dimension, then each line will be considered as an entity (node). The bottom left window displays the computed taxonomy of dimensions. Orange coloured nodes represent dimensions that can be used as id for entities. The bottom right window displays the hierarchy of dimension obtained by stretching out the taxonomy.

If we consider the address example in Table I. We obtain the following ranking (hierarchy): ”Continent” $\geq$ ”Country” $\geq$ ”City” $\geq$ ”Name”. So if two people (rows) in the table live in the same city, for example Paris, obviously they live in the same country, France and the same continent, Europe. So in the graph there will be three edges, one for the city matching, one for the country matching, and one for the continent matching. This illustrates that there will be a lot of edges in the graph. So in order to limit this number of edges, we decide to add edges according to the bottom level dimensions in the taxonomy, and add information on these edges. For example, instead of having three edges in the previous example, there will be one edge having three attributes. One signalling that this edge can be consider for relations according to ”Continent”, one signalling that this edge can be consider for relations according to ”Country” and one last for signalling that this edge can be consider for relations according to ”City”. So the number of edges will strongly decrease. But as each edge can define relations through several dimension, it will be difficult to understand according to which dimension entities are in relation. So we set a filtering system which will be describe in Section IV.

C. Complexity

Our method enable users to find a taxonomy of dimensions and build a graph according to this taxonomy. But explore the entire space of dimensions combinations in order to highlight patterns between them is not conceivable in term of complexity. In fact, if we have a table of n rows and m dimensions, compare two dimensions is an operation in $\mathcal{O}(n^2)$. And if we do that for all the possible couple of dimensions we have finally a complexity of $\mathcal{O}(m^2 \times n^2)$ if we try to compute the entire taxonomy. This complexity can be seen as $\mathcal{O}(\min(n, m)^4)$.

So we have to limit the number of compared dimensions. The more efficient way is to ask the user for selecting dimensions he would like to know something about. We describe now how we solicit the user.

D. User/expert selection

Due to the size of the considered tables and due to time complexity of the algorithm one needs to apply, asking to the end user to manually select the set of dimension he/she wants to consider is necessary. We have shown that the ”limiting” factor of the complexity is the minimum between the number of rows and the number of dimensions in the table. In most cases the number of dimensions is lower than the number of rows. So decreasing the number of dimensions to treat, decrease significantly the complexity of the taxonomy’s computation. So taking advantage of the user experience increase the performances of our method. In order to collect information about which dimensions the user want to analysis, we propose him a spreadsheet view of the entire table. In this view, it is possible to check or uncheck dimensions. Only checked dimensions will be kept for the computation. Discarding dimensions can cause to reduce also the number of rows in the table. If two rows only differ on one dimension and that dimension is discard by user, keeping the two rows will not give more information to the user.

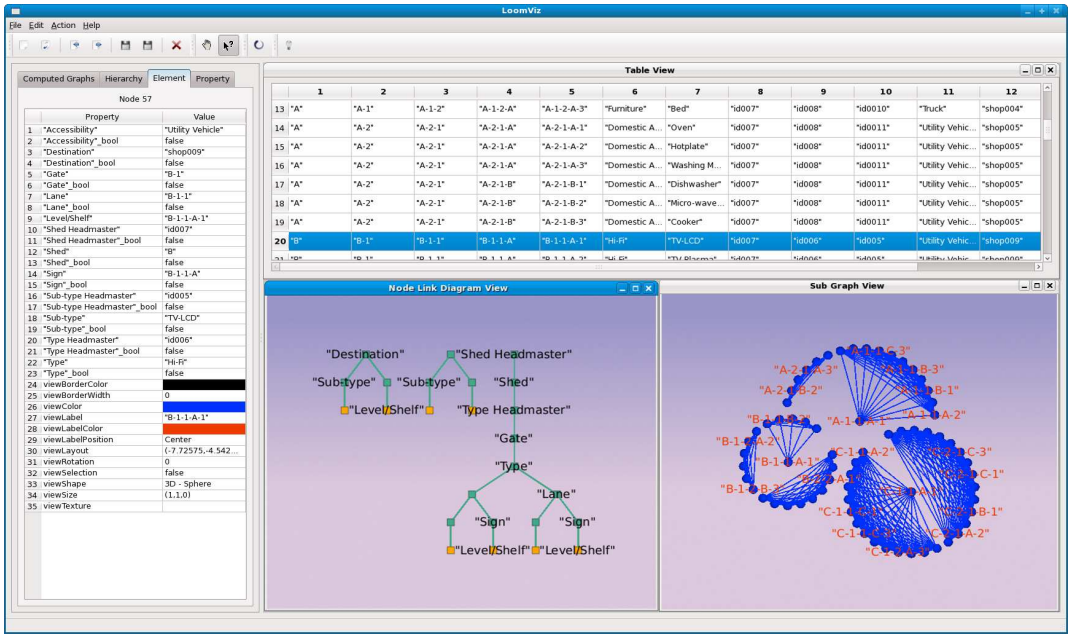

Figure 4. The complete system. The ”Table View” window displays a table view of the data. The ”Node Link Diagram” window displays the stretched taxonomy (as seen in Figure3). The ”Sub Graph View” window displays a subgraph pointed by the hierarchy. On the left, the table in the ”Element” tab displays the properties of a node clicked in the ”Sub Graph View”. The row corresponding to the node is highlighted in the ”Table View”.

IV. Visualization: Filtering edges through the taxonomy

As we add edges between all the entities that share a value on one dimension, there will be a lot of complete induced subgraphs (if we consider only few vertices V' of the graph, each vertex of V' is connected to all other vertices of V'). And as one edge can be used for many relations, one needs to be able to filter edges to only show up the relevant ones. For that purpose, we build a hierarchy of subgraphs. This hierarchy correspond to the taxonomy of the ranked dimensions. Each branch of the hierarchy is a path in the taxonomy so finally we have unfolded the taxonomy.

Each node of a hierarchy's branch corresponds to a dimension. This node is linked to a graph that represents the subgraph obtained by filtering the edges according to the dimensions placed above it in the hierarchy. For example, let consider the branch ”Continent” $\geq$ ”Country” $\geq$ ”City” $\geq$ ”Name”. The subgraph linked to ”City”, will contains all the edge that satisfy a relation through ”City”, but also through ”Country” and ”Continent”.

V. Case study

The dataset which is used here describes products that are stocked in sheds, it is composed of thirteen dimensions. Some are used to localize the products, as: ”Shed”, ”Gate”, ”Lane”, ”Sign”, ”Level/Shelf”. Some are used to describe the products: ”Type”, ”Sub-type”. There are also information about people who manage the products: ”Shed Headmaster”, ”Type Headmaster”, ”Sub-type Headmaster”. And some other information such as ”Accessibility” which defines how products can be extracted from the sheds, and ”Destination” which gives information about the shop that owns the products. Each row of the dataset collects all these attributes.

The method developed here has been implemented using Tulip [11]. Tulip is a software providing tools and graphic components for data visualization and more precisely graph visualization.

The first step of the method consists in selecting dimensions in the table that will be taken into count for the taxonomy and graph generation. As we don't know anything about this dataset, we compute the taxonomy and the graph using all the dimensions of the table. The taxonomy of dimensions that we obtain is the taxonomy shown in the Figure 3,and the stretch out taxonomy is the hierarchy of dimensions shown in Figure 3 and also in Figure 4.

Then we look at the taxonomy and the hierarchy in order to bring out information about which subgraphs will be interesting to study. First, we can see that there are three sub-hierarchy in that taxonomy, one rooted on ”Destination”, one rooted on ”Accessibility” and one rooted on ”Shed Headmaster”. We can see that ones rooted on ”Destination” and ”Accessibility” are very similar. So subgraphs generated from these two sub-hierarchy will be similar, and this fact allows us to notice that we will not have to study all the generated graphs but only the half of these graphs (one of the two sub-hierarchy). On the other side, the third part of the hierarchy hold a long branch, that tells us that these

dimensions are ranked. We also know that down dimensions give more precise relation that the top ones, so "Sub-type" and "Sign" dimensions give the precisest relation between entities, whereas "Shed Headmaster" dimension give the more general relation. Graphs generated by "Shed Headmaster" will have a lot of edges and will not give so much information, whereas graphs generated by "Sign" or "Sub-type" will as less edges.

The last thing we can say about the taxonomy/hierarchy computation concern the orange coloured nodes. We know that these orange coloured nodes represent dimensions that can be used to identify identities. It is not useful to look at the graphs generated by these dimensions. In fact, if a dimension identifies entities, that means that entities have different values according to that dimension. So, as we add edges to a graph only if some entities have a same value, the graphs generated will have no edge.

In Figure 4, the "Sub Graph View" window displays the graphs obtains according to the "Type" dimension. In this graph we can see seven connected components. Each connected component corresponds to one value of the "Type" dimension, so each connected component is a cluster of the entities. So studying the connected component, we can observe how dimensions cluster the entities. In the "Type" graph, we have a Furniture products cluster, a Domestic Appliance cluster, a Hi-Fi cluster, a Computer cluster, a Media cluster, a Do-It-Yourself cluster and a Clothes one.

Now if we want to have a look to all the attributes of an entity, we just have to click the node in the graph. As we can see in Figure 4, the row corresponding to the clicked node is highlighted in the "Table View" window. Attribute of a node are also displayed in the "Element" tab on the left of the system. In Figure 4 the node identified by "B-1-1-A-1" has been clicked and the corresponding row is highlighted in the "Table View" and all its attributes are displayed in the "Element" tab on the left.

Using the hierarchy of dimensions, it is possible to visualise how the ranking of the dimension is effective on the sub-graphs generation. Clicking on successive nodes of a branch of hierarchy starting from top, will display them one after one. And as a graph associated to a dimension is a subgraph of its father graph, iterating on graphs will display how clusters evolve through the hierarchy of dimensions. As the layout of the node don't change, we will be able to easily see edges disappearing for top to bottom exploration and appearing for bottom to top exploration.

VI. Conclusion

We presented a method for automatically generate graphs from a dataset in tabular format. That method enables to ease the data transformation task necessary to use graph based Infovis Systems. We described an algorithm that automatically generates taxonomy of dimensions. Using both that taxonomy and a hierarchical graph based exploration tool, we are able to provide to end user a system that enable to interactively explore the set of possible data transformation. We demonstrated the usefulness of our solution with a complete case study.

References

[1] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymki, and K. Wenger, "Devise: Integrated querying and visual exploration of large datasets," in *In Proceedings of ACM SIGMOD*, 1997, pp. 301–312.

[2] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: a system for query, analysis, and visualization of multidimensional databases," *Commun. ACM*, vol. 51, no. 11, pp. 75–84, 2008.

[3] J. D. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," vol. 13, no. 6, 2007, pp. 1137–1144.

[4] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[5] Landauer, *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[6] M. Smith, B. Shneiderman, N. Milic-Frayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, and E. Gleave, "Analyzing (social media) networks with nodexl," in *C&T '09: Proceedings of the fourth international conference on Communities and technologies*. New York, NY, USA: ACM, 2009, pp. 255–264.

[7] A. Slingsby, J. Dykes, and J. Wood, "Configuring hierarchical layouts to address research questions," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 977–984, 2009.

[8] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon, "Manyeyes: a site for visualization at internet scale," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1121–1128, November 2007.

[9] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *VIS '91: Proceedings of the 2nd conference on Visualization '91*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 284–291.

[10] C. Ware, *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[11] D. Auber, "Tulip : A huge graph visualisation framework," in *Graph Drawing Softwares*, ser. Mathematics and Visualization, P. Mutzel and M. Jünger, Eds. Springer-Verlag, 2003, pp. 105–126.